

Miika Lumme

**MOBILE PLATFORM SENSORS FROM A PROGRAM-
MER'S POINT OF VIEW**

Thesis

CENTRIA UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Information Technology

April 2015

ABSTRACT

Unit Kokkola-Pietarsaari	Date April 2015	Author/s Miika Lumme
Degree programme Information Technology		
Name of thesis MOBILE PLATFORM SENSORS FROM A PROGRAMMER'S POINT OF VIEW		
Instructor Sakari Männistö		Pages 44
Supervisor Sakari Männistö		
<p>The purpose of this thesis was to study sensor interfaces available for programmers in the three major mobile platforms. These platforms include Android from Google, iOS from Apple and Windows Phone from Microsoft. This research was done from programmer's point of view.</p> <p>Firstly, all of the sensors embedded in nowadays mobile devices were introduced, and then the sensors that are connected wirelessly were introduced. Research was done in what is embedded in the mobile devices and how a programmer could program any of these sensors in any of the mobile platforms.</p> <p>Secondly the research comes to the comparison part. In this part all of the three mobile platforms were compared in many different areas that programmers would use and need. Firstly, comparison was done in the API of these mobile platforms. Afterwards came the comparison of one of the most needed aspects for programmer, the programmer guidance for the different mobile platforms. Finally the comparison between the usability of all these three mobile platforms was done.</p> <p>Conclusions state that from a programmers view point Android is the best mobile platform, but the easiest one to use is Apple. Windows phone is in the middle grounds between these two even the programmability of these platforms stay in the same order, starting from Android.</p>		
Key words Android, iOS, mobile platform, sensor, Windows phone		

ABSTRACT

LIST OF ABBREVIATIONS

CONTENTS

1 INTRODUCTION	1
2 GENERAL INFORMATION ABOUT MOBILE SENSORS	3
2.1 Accelerometer	4
2.2 Motion detection sensor	4
2.3 GPS	5
2.4 Camera	5
2.5 Microphone	6
2.6 Ambient light sensor and proximity sensor	6
2.7 Magnetometer	7
2.8 Near Field Communication and radio frequency identification	8
3 WIRELESSLY CONNECTED SENSORS	9
3.1 Bluetooth	9
4 DIFFERENT MOBILE PLATFORMS AND SENSOR PROGRAMMING	11
4.1 Android	11
4.1.1 Sensors in Android	11
4.1.2 Sensor programming in Android	13
4.1.2 Sensor programming in Android API	17
4.2 iOS	21
4.2.1 Sensors in iOS	21
4.2.2 Sensor programming in iOS	22
4.3 Windows Phone	26
4.3.1 Sensors in Windows Phone	27
4.3.2 Sensor programming in Windows Phone	27
5 DIFFERENCES IN THE MOBILEPLATFORMS	30
5.1 Difference between the API	31
5.2 Difference between programmability	32
5.3 Difference in designer/developer guidance between platforms	37
5.4 Difference between usability	39
6 CONCLUSION	41
REFERENCES	43

1 INTRODUCTION

Nowadays, almost every smart mobile phone has their own sensors built in them. People use their own smart phones on daily basis and might not actually think or know that they have so many sensors built in them. Most of today's smart phones have at least 6 sensors in them. Two of the most commonly used sensors are microphone and camera. Microphone and camera are not included in the API's and frameworks as sensors, but they have been studied in this thesis because they work like a sensor and act like a one, and they are used like any other sensors. They are even regarded as one in the most general context. People might not think of them as a sensor, because they are used so often and regularly. The more common sensors include for example ambient light and proximity sensors that are also used daily by smart phones, but people are not aware of that.

Smart phone sensors are also used to gather a lot of information. As an example of this is the fitness applications such as Sports Tracker they use motion data collected by motion sensor to find out the speed at which the person is moving and position data that then shows the position or track what the person has taken. Sports Tracker can also use external devices to help it gather even more information through sensors, such as heart rate monitor belt which will then give the information of the person's heart rate through a series of sensors and data collected by them.

This thesis work is about the mobile platform sensors, but the view of the mobile platform sensors is from programmer's point of view. How sensors can be programmed for each different mobile platform? This is done for these three mobile platforms: iOS, Android and Windows Phone. This thesis compares each of these mobile platforms, their programmability and how sensors are used in them and compares how things differ in another API. In this thesis there is also research work about related mobile platform sensor technologies. The results and conclusions of this thesis are that from a programmers view point Android is the best one in either programming or usability, but for a normal person Apple phones are the easiest ones to

use. Windows phone is in the middle grounds in this because it is not the easiest one to program nor is it the easiest one to use for beginners.

2 GENERAL INFORMATION ABOUT MOBILE SENSORS

Before smartphones, people would only interact with a small range of narrowly focused sensors in their daily life. Usually these sensors which they interacted with resided in a single device and these sensors were usually designed only for single purpose (oven temperature sensors, tire pressure sensors, television remote control systems). When smartphones were introduced an exciting range of different sensors became available to users and developers of smartphones. Before smartphones, sensors rarely existed in such quantities or in such proximity to the user. The availability of multiple sensors in a single device added a wide array of uses for the phone, the amount of sensors included in smartphones can be seen from GRAPH 1. (Milette & Stroud 2012.)



GRAPH 1. iPhone 4 showing eight different sensors that almost all smartphones have. (Lane, Miluzzo, Lu, Peeble, Choudhury & Campbell 2010.)

2.1 Accelerometer

Accelerometers have become common thing in smartphones after they had been introduced to enhance the user interface and the use of camera in mobile phone. Accelerometers are used to determine the orientation in which the phone is held by the user and to use that information to display the display in such a way that it will either orientate the screen into landscape or portrait view depending on which the user is holding it in. (Lane, Miluzzo, Lu, Peeble, Choudhury & Campbell 2010.)

Accelerometer sensor refers to an absolute orientation with respect to the Earth. Accelerometer is a raw three-axis inertial sensor and accelerometer reports values corresponding to the devices coordinate system. The coordinate system of a smartphone is partially defined by default orientation; this differs in each type of device. As an example phones have portrait default orientation and tablets have landscape default orientation. When a device is in its default orientation, the axes are usually directed in such a way that the x-axis is horizontal with positive values to the right, y-axis is vertical with position values upward and the z-axis is positive values in front of the screen. The axis orientations in the device do not change when the orientation changes, it will remain the same. (Milette & Stroud 2012.)

2.2 Motion detection sensor (Gyroscope)

Gyroscopes use tiny masses on tiny springs to measure the rotation between x, y and z-axis. They are designed to measure the Coriolis force due to rotation. The Coriolis force is the tendency for a free object to veer off course when it is viewed by a rotating reference. The values are reported usually in radians per second. Gyroscope is implemented in MEMS-technology. (Milette & Stroud 2012.)

2.3 GPS

The Global Positioning System (GPS) uses a system of satellites that are orbiting the planet to help a mobile phone to determine its current location. GPS uses the information from GPS satellites that are orbiting the Earth to triangulate its position. Today, GPS system consists of 27 satellites that are continually orbiting earth and transmitting information to receivers. For GPS to be able to calculate the location of the mobile phone, it must be able to determine its distance from multiple satellites. GPS uses ephemeris data to do this. By using the distance that multiple satellites will give the mobile phone, it will be able to triangulate its current location. Minimum of three satellites are required for GPS to triangulate the position correctly. (Milette & Stroud 2012.)

GPS can also be used in combination with other sensors for example accelerometer to find out the speed at which a person is moving and where. With the information combined from the accelerometer and the data from GPS, the GPS will be able to determine what kind of transportation a person is using, for example if he is using a car or a bicycle or something even more faster. This is done by the GPS in such a way that it will find out the speed at which the person is moving via satellite data. (Lane, Miluzzo, Lu, Peeble, Choudhury & Campbell 2010.)

2.4 Camera

Camera to mobile phones started appearing in 2001, and now they are everywhere, every single Smartphone has one and they are used a lot by mobile phone users. The camera in mobile phone these days are used for example photo blogging with today's social media, for example Instagram, Twitter and Facebook. Camera can also be used for more specialized sensing, for example activities such as tracking the users eye movement across the display of mobile phone as means to activate an application which they are looking at. (Lane, Miluzzo, Lu, Peeble, Choudhury & Campbell 2010.)

2.5 Microphone

Many applications use microphone to provide some kind of audio recording or for speech recognition on all the mobile platforms iOS, Android and WP. There are some applications that allow the user to create audio notes and also with today's Multimedia Messaging Service (MMS) you can send audio files to anybody you want to by just recording your speech via microphone. Usually these saved audio files are also automatically saved in .wav format so you can use and open them with many different applications and programs with ease. (Allan 2011.)

2.6 Ambient light sensor and proximity sensor

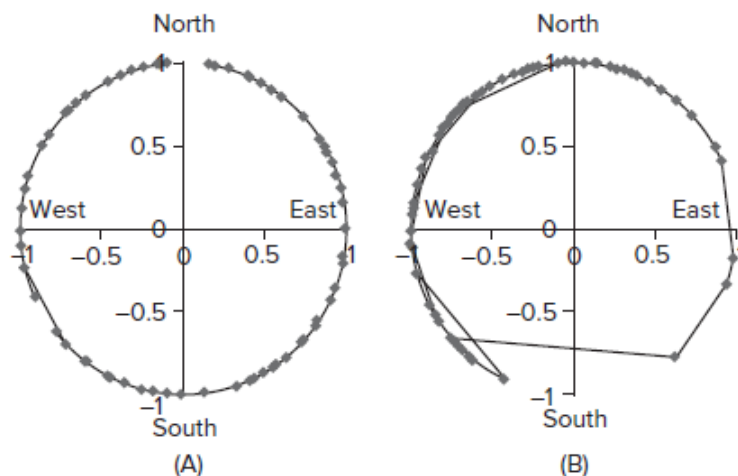
Ambient light and proximity sensor allows smartphones to perform context recognition tasks with association of the user interface. Proximity sensor is used mainly for the purpose of recognizing when the user of the Smartphone holds the phone to his/her ear and shuts the screen off and disables the keys. This prevents the user from accidentally doing something with the touch screen when he/she is talking on the phone; it also saves power because the screen is turned off. (Lane, Miluzzo, Lu, Peeble, Choudhury & Campbell 2010.)

Ambient light sensor is only used for either brightening or darkening the screen display. The light sensor will automatically check the lightness in the room and will try to balance the screen light with that, so that the user's eyes won't be damaged by either the brightness or darkness of the screen. Also the user experience will be a lot nicer after the automatic light balancing. (Zhou, Zhu, Zheng & Yang 2011.)

2.7 Magnetometer

The magnetic field sensors in smartphones may operate under a variety of different methods; this depends on the manufacturer and the architecture of the smartphone. They use Hall effect, magneto-resistive materials or the Lorentz force. Hall effect sensors have currently the largest market share of magnetometers and they work in the most efficient way, it only needs to simply pass a current through a wire. Then a magnetic field component perpendicular to that wire causes the electrons to have higher density on one side than another of the wire. This results in a voltage across the width of the wire that is proportional to the magnetic field. Lorentz force sensors are almost similar, but they measure a mechanical deflection of the wire rather than the voltage across the wire's width. (Milette & Stroud 2012.)

Even with today's magnetic field measuring systems you may notice that the magnetic field readout is quite jumpy and may seem less accurate than other sensors in smartphones. Also because magnetic field sensors can be influenced a lot by nearby metal, some people have used the sensor to make for example an Android device into a crude metal detector. GRAHP 2 shows us the interference that is caused by nearby metal objects when using magnetometer. (Milette & Stroud 2012.)



GRAPH 2. (A) Smartphone rotated at approximately constant rate, magnetic sensor readings. (B) Device rotated in the same way, but with metal object nearby. (Magnetic field disturbance) (Milette & Stroud 2012.)

2.8 Near Field Communication and radio frequency identification

Radio frequency identification (RFID) tags can be acquired in many different factors. Some of these are cards, key fobs, but the most common one to date is RFID sticker. The RFID sticker has rectangular coil of metal strips in it, these coils are used as antennas for the sticker so that it can listen for any radio frequency. Within these coils are other large blocks, these blocks are integrated circuits which are made of silicon. GRAPH 3 shows us the size and structure of RFID sticker. These integrated circuits can store a small amount of data in them and send the information back to the RFID reader via the antennas. (Milette & Stroud 2012.)



GRAPH 3. Internal components of RFID sticker. (Milette & Stroud 2012.)

Near Field Communication (NFC) share the same basic technology with RFID, but the biggest difference between these two is that NFC is always going to be a short range scanner, just like the name says. The range of NFC is going to be between 1–4 cm. Another difference between the NFC and RFID is the size of the data transmission that they both can send. RFID contains only 40-bit unique identifier and these are read-only data. Whereas the NFC tags can usually store up to 48 bytes of data, which average in about 144 bytes and might even go up to 8 kilobytes for the largest of the NFC tags. Also the data in NFC can be both rewritten and read by a reader if the NFC tag is not write-protected. (Milette & Stroud 2012.)

3 WIRELESSLY CONNECTED SENSORS

Nowadays, even though mobile devices do have a lot of sensors embedded in them, there are still quite a few sensors that are connected to mobile devices either wirelessly or with a wire. Examples of these are wearable sensors, these are worn by people and they gather data and then send it either via Wi-Fi or Bluetooth to the mobile device. One example of these is heart rate monitor. Wi-Fi and Bluetooth have both given so much more availability for sensors to connect to mobile devices without actually being connected to the mobile device via wire. (Aram, Troiano & Pasero 2014.)

There are some environmental sensors that can be used to collect data, for example temperature or humidity sensors. These sensors usually connect to the mobile phone via short distance transmitted data. This is done usually to accomplish the Low power and low cost that are the main characteristics of these type of sensors. Usually Bluetooth is applied to these kinds of sensors as the receiver, this is because it allows the mobility of the receiver. Also almost every mobile phone has Bluetooth embedded in them. (Aram, Troiano & Pasero 2014.)

3.1 Bluetooth

Bluetooth has been used now for almost 15 years. It has been used in many different technologies, nowadays mobile devices have it embedded in them. Bluetooth is one of the most used ways of connecting devices together; this is because it is based on a low cost and low power, short range radio technology. Bluetooth was originally developed just to connect devices which needed cable replacements, such as wireless mouse, headphones, and portable computers and so on. The availability of Bluetooth has created the notion of Personal Area Network, which is close range wireless network. (Bray & Struman 2001.)

The specification that Bluetooth uses is open global specification that is defining the complete system from the radio right up to the application level. The protocol stack is almost always implemented partly in the hardware and partly in the software that is running on a microprocessor. It runs with different implementations that are partitioning the functionality between the hardware and software in many different ways. (Bray & Struman 2001.)

When Bluetooth is chosen for wireless communication systems, it is usually done because Bluetooth has relatively low power consumption. This is compared to the power consumption of other high data rate wireless communication systems such as Wi-Fi. One other reason also is the simplicity of Bluetooth and the last reason is the width at which Bluetooth is used worldwide. As Bluetooth is embedded in every Smartphone, it is usually preferred when talking about mobile platform wireless connections for any kind of sensor. (Aram, Troiano & Pasero 2014.)

If a programmer wants to program an app, which is going to use Bluetooth for any of the three following platforms: iOS, Android or Windows Phone. He needs to first check if there are any limitations of programmability of Bluetooth. For example Apple has really strict way in which they work, so they have limited the use of hardware for programming especially with the ports that are related to Bluetooth. Apple only allows commercial program to communicate via Bluetooth with non-Apple based devices. Whereas Android is really different. Regarding to Bluetooth everything is open, it is easy to customize and really open to work with any different hardware through Bluetooth. (Aram, Troiano & Pasero 2014.)

Bluetooth LE has couple key advantages over similar technologies in such a way that it will be able to build onto the existing Bluetooth infrastructure that is nowadays in almost every mobile device, and laptop. When one of these devices gets upgraded to use LE chips, they can act just like a gateway for any LE sensor. Therefore the LE is expected to be one of the most significant contributors to overall wireless markets. This is very true in the case of wireless sensors. (Gupta 2013.)

4 DIFFERENT MOBILE PLATFORMS AND SENSOR PROGRAMMING

The three dominant mobile platform providers are Apple, Google and Microsoft. Apple started its rise in 2007, Apple's mobile platforms are used in their own iPad's and iPhone's and their mobile platform is called iOS. Google has their own mobile platform named Android which is most commonly used in Samsung's smartphones, but also used in phones like HTC. The last one of these three companies is Microsoft, which is well known for their computer operating systems. Nowadays they have their own Windows Phone, which uses Windows as the mobile platform.

4.1 Android

Android does combine ubiquity of cell phones, open source software, and android also has the corporate backing of Google and other Open Handset Alliance members, such as Motorola, HTC, Verizon and AT&T. Starting programming with Android is quite easy. Person do not even need to have own an Android device, because the Android SDK which is used for programming Android devices has its own emulator which they can use if they do not have their own Android device. (Burnette 2010.)

4.1.1 Sensors in Android

The availability of sensors in Android is one of the features what makes them different from any other computer. Without any sensors, Android phone would be just a phone which can be used to browse web and has a screen which is too small. Sensors in Android phone allow the phone and apps do amazing things. As an example of this is that the sensors can help users to do tasks that wouldn't be possible without them. Because of all of this it is most likely essential for an app to use sensors or at least incorporate them if it wants to be successful. Sensors will keep on being an essential part of Android phones because the hardware in these phones is improving all the time, which makes the number of sensors and their quality to increase as well. (Milette & Stroud 2012.)

Android's capability to use sensors and sense with them is possible only because of the available hardware that is on Android phones. A capability may either use values directly from one of the built in sensors in Android phone, such as magnetic field sensor. Android phone also can use combination of given devices when collecting information, such as the camera and microphone with each other. Capability could also use combination of server based data and hardware included in it, a good example of this is speech recognition. This uses both, hardware (microphone) and the server based data (recognition of the speech, language etc.). All of this can be then used as data to inform an app about the state of the device. (Milette & Stroud 2012.)

Android devices have many different physical sensors built in them. Almost all the sensors that have been built in Android phones are microelectromechanical sensors (MEMS), these are small tiny scale sensors made of silicon chips. MEMS use techniques that are borrowed from computer-chip manufacturing. MEMS sensors usually are noted as the sensors that move inside the device, such as the pressure sensor, accelerometer, gyroscope, and possibly compass. These are the true MEMS sensors. (Milette & Stroud 2012.)

When referring to sensors through Sensor class it can have two different types. These two types are: a raw sensor or a synthetic sensor. Raw sensor is a sensor which gives only the raw data from any sensor, and usually when talking about raw sensors, the sensor is an actual physical sensor built inside the Android device. Whereas synthetic sensors are sensors that will most likely combine raw data from two raw sensors, or another way is that the synthetic sensor will modify the data of a raw sensor in such a way that it is easier to compute. TABLE 1 shows the synthetic and raw sensors. When a programmer wants to access any type of sensor, regardless of the sensor type it can be done same way of using the sensor API of Android. (Android Developer 2014.)

Raw sensors	Synthetic sensors
<ul style="list-style-type: none"> •Sensor.TYPE_LIGHT •Sensor.TYPE_PROXIMITY •Sensor.TYPE_PRESSURE •Sensor.TYPE_TEMPERATURE •Sensor.TYPE_ACCELEROMETER •Sensor.TYPE_GYROSCOPE •Sensor.TYPE_MAGNETIC_FIELD •Sensor.TYPE_RELATIVE_HUMIDITY •Sensor.TYPE_AMBIENT_TEMPERATURE 	<ul style="list-style-type: none"> •Sensor.TYPE_ROTATION_VECTOR •Sensor.TYPE_LINEAR_ACCELERATION •Sensor.TYPE_GRAVITY •Sensor.TYPE_ORIENTATION

TABLE 1. Type of sensors. (Milette & Stroud 2012.)

4.1.2 Sensor programming in Android

GPS is nowadays one of the most used sensor in phones. When you know your device's current location, it will enable the developers of the applications to increase the functionality to wide range of apps. Location is one of the key components for some of apps such as Google Maps, Google Navigator and location is also used today in some other apps such as Twitter, and Facebook. When you are working with Android and its location services, you need to have a look what tools you have to work with. This is quite easy for Android, this is because the majority of the classes that you will need are located in android.location package. These are the classes in the location package as showed in FIGURE 4. (Milette & Stroud 2012.)

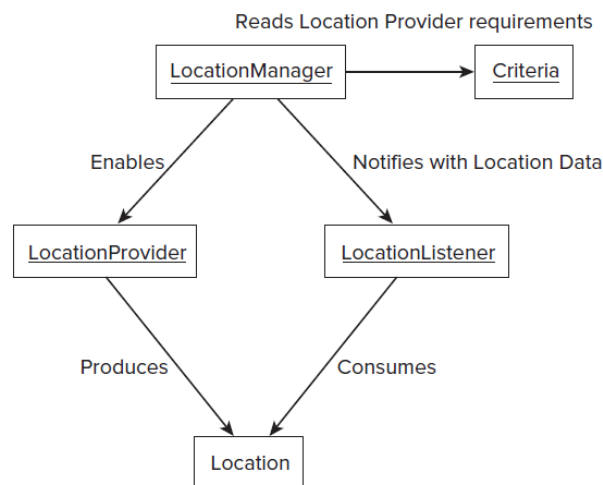


FIGURE 4. Android location components. (Milette & Stroud 2012.)

LocationManager (LM) is the main point of entry when using location services in Android. LM is the part which allows an app to tell Android when it is necessary to update the location information and when it is not needed anymore. LM also provides information about the current state in which the location system is, such as GPS status information and enabled location providers. LM can also give you the last known location in which the device has been. LocationProvider is an abstraction for most of the different sources of any information in Android. Android provides many different sources of location data; this data has many different characteristics in them which are drastically different. Even when each of these providers will generate location data in their own different way, they will all in the end communicate with an app the same way and they will also provide similar data to the app in the same way. (Milette & Stroud 2012.)

Location class is the part that actually encapsulates the actual location data provided to any app from a location provider. This data will always contain the quantifiable data, which are latitude, longitude, and altitude. When an app has received all the necessary data from Location class, it will start the application-specific processing of that data. One really important thing to note when using Location class is that even though it has all the properties for a wide range of location data, not all of the location providers will give all necessary values for every property. An example of this is if an app uses a location provider which for some reason does not use or provide altitude, then the location data given will not include altitude or any information about it in the data. Location class also provides methods which would allow in this case checking if any altitude has been used. (Milette & Stroud 2012.)

Criteria class can be used by apps to query LM for a certain characteristics including location providers. Criteria class is useful when an app is not concerned about the actual location providers, but instead of that it is concerned about the same characteristics which some location providers could have. Once Criteria class is instanced to an app, this app can then set/unset attributes on a Criteria class with which they can reflect the characteristic of any location provider that it is interested in. (Milette & Stroud 2012.)

LocationListener (LL) contains a group of callback methods, these methods are called when the device changes its current location or when the device's location service has changed. The LM usually have enabled an app to either register/unregister a LL implementation which can be then used by the app to track the changes that have been made on the device. The two most commonly used methods to receive any location updates from location services are either LL or a PendingIntent. (Milette & Stroud 2012.)

There are couple location providers in Android that are used, all of them give varying level of accuracy but these location providers also have varying level of battery consumption. Each one of these could have a big impact on the overall use of an app. These providers that Android have are: GPS location provider, Network location provider, and Passive location provider. Usually an app choose which one of these to use either by explicitly registering each location provider that the programmer wants to use in LM, or another way to do this is by setting attributes in the Criteria object and then passing this object to the LM. The more useful way to do this is by using Criteria, this allows the user to customize the source from which the location data will come from. This could be quite important to some users, because use of some of the location providers such as Network location provider can cost them money. (Milette & Stroud 2012.)

GPS location provider works in such a way that it will communicate with satellites orbiting Earth and time to find out the current location of the device. GPS tends to be the most accurate location data that can be found, however the GPS location provider will use more battery power than any other location provider, this is because GPS uses an separate radio to find out the location, this will then drain the battery more quicker than any other location provider. Even sometimes GPS could take a while to fix the location of the device. The network location provider uses only two data sources with which it will then find out the location of the device. These are Wi-Fi network location and cell-tower location. The time to first fix (TTFF) for network location provider can be a lot smaller than for GPS, but the network provider gives much less accurate location data to the device. (Milette & Stroud 2012.)

Then as the last location provider, the passive location provider which allows any app to receive its location information without having to request the location information from LM. The passive provider will use other apps requests of location data in such a way that then another app has requested for location data and it has been sent to the app, this will then allow the passive provider to piggy-back on the information that has been sent to other apps and find out that way the location of the device, and the user doesn't have to make any special request for location data. (Milette & Stroud 2012.)

The most common battle when using location services is that do you want to use either more battery life or get more increased accuracy for your location. Although almost every app could use the more accurate data, it is not always necessary for them, especially by the expense of battery power. The battery consumption amount for different type of apps is shown in TABLE 2. (Milette & Stroud 2012.)

LOCATION PROVIDER	REQUIRED PERMISSION	BATTERY CONSUMPTION	ACCURACY
GPS Provider	<code>android.permission.ACCESS_FINE_LOCATION</code> or <code>android.permission.ACCESS_COARSE_LOCATION</code>	Consumes more battery power than other location providers	Provides the most accurate location data
Network Provider	<code>android.permission.ACCESS_COARSE_LOCATION</code>	Consumes less battery power than the GPS provider	Provides less accuracy than the GPS provider
Passive Provider	<code>android.permission.ACCESS_FINE_LOCATION</code>	N/A	N/A

TABLE 2. Location Providers and their battery consumption vs. accuracy table. (Milette & Stroud 2012.)

LocationManager is usually the front door to the location services on an Android device, and the app needs to get reference to it at some point. This is done by using the `Activity.getSystemService(LOCATION_SERVICE)`. Most commonly this is done in the `onCreate()` method because of the multiple calls that are going to be made to LocationManager throughout the Activity lifetime. Because the `onCreate()` method is the very first method that is going to be called in any activity's lifetime, it is required to acquire the location manager references

here at the start. GRAPH 5 will show how you can implement the onCreate() method to your Activity. (Milette & Stroud 2012.)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.current_location);

    locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);

    latitudeValue = (TextView) findViewById(R.id.latitudeValue);
    longitudeValue = (TextView) findViewById(R.id.longitudeValue);
    providerValue = (TextView) findViewById(R.id.providerValue);
    accuracyValue = (TextView) findViewById(R.id.accuracyValue);
    timeToFixValue = (TextView) findViewById(R.id.timeToFixValue);
    enabledProvidersValue = (TextView) findViewById(R.id.enabledProvidersValue);
}
```

GRAPH 5. Obtaining reference to the LocationManager. (Milette & Stroud 2012.)

Location of the Android device can also be used to track the device everywhere it moves, this requires the use of GPS or Wi-Fi providers continuously. This requires the collection and persistent location data to be sent to the app. It should continue collecting the data no matter what is done with the phone, until the app is either turned off or the phone is turned off. There are two main parts and applications that are used for this. These tasks are services and broadcast receivers. When using the continuous location of the Android device, it will drain quite a lot of the battery power, so it is not recommended to use this too much. (Milette & Stroud 2012.)

4.1.3 Sensor programming in Android API

Android Sensor API consists of all the classes that are required for requesting and processing the information given by sensors that are built in the device's hardware. The first entry point to the Sensor API is the SensorManager class. This class allows any app to request or sent any sensor information. Once the data is registered, data values will be sent to another part of the API which is SensorEventListener, and the information is sent here in the form of SensorEvent. These are the three parts of the Android Sensor API that are required to get information from a Sensor. (Android Developer 2014.)

SensorManager is part of the API that allows you to access your device's sensors. The sensor data can be collected after the sensor has been registered. One instance of this class is for example Context.getSystemService() with argument SENSOR_SERVICE. GRAPH 6 shows us an example code of how to get accelerometer data from a built in sensor by using SensorManager. (Android Developer 2014.)

```
public class SensorActivity extends Activity, implements SensorEventListener {
    private final SensorManager mSensorManager;
    private final Sensor mAccelerometer;

    public SensorActivity() {
        mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        mAccelerometer =
mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    }

    protected void onResume() {
        super.onResume();
        mSensorManager.registerListener(this, mAccelerometer,
SensorManager.SENSOR_DELAY_NORMAL);
    }

    protected void onPause() {
        super.onPause();
        mSensorManager.unregisterListener(this);
    }

    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }

    public void onSensorChanged(SensorEvent event) {
    }
}
```

GRAPH 6. Example code how to use SensorManager. (Android Developer 2014.)

SensorEventListener is a listener that is used for receiving notifications from SensorManager when any sensor value has changed at all. Some ways to do this SensorEventListener is through its public methods which are: onAccuracyChanged(Sensor sensor, int accuracy) or onSensorChanged(SensorEvent event). If either one of these is going to happen, the SensorEventListener is going to trigger. (Android Developer 2014.)

SensorEvent is the data structure that contains all the information that is needed or is going to be passed to an app when any hardware sensor in the device has any information to report.

SensorEvent object is related to SensorEventListener in such a way that when SensorEvent object is going to be passed to callback methods in SensorEventListener. The listener will then process the object given to it in an application-specific manner. There are few data members that are included in SensorEvent. These data members are: SensorEvent.accuracy, SensorEvent.sensor, SensorEvent.timestamp, and SensorEvent.values. (Milette & Stroud 2012.)

When you want to determine the orientation of your device, which is somewhat useful in almost all of the possible apps out there, especially games. When a human changes the orientation of the device, the app will follow in its path and change the orientation via the sensors that are built in the device. The sensors that actually do check the orientation of the device are gravity sensor, accelerometer and magnetometer, and vector synthetic sensor as stated previously. For example you can use the gravity sensor to determine the orientation of the device in such a way that it will calculate the gravity applied to the X, Y and Z axes in the zeroth, first and second slots in an array, respectively. Z axis is the one that goes through the screen of the device and out of the back of the device, the code needs to use exactly the third (offset2) value in the array of X, Y and Z. Here is an example code for how to determine the orientation of the device by using gravity sensor. (Milette & Stroud 2012.)

```
private static final double GRAVITY_THRESHOLD =
    SensorManager.STANDARD_GRAVITY / 2;
...
case Sensor.TYPE_GRAVITY:
    ...
    if (event.values[2] >= GRAVITY_THRESHOLD)
    {
        onFaceUp();
    }
    else if (event.values[2] <= (GRAVITY_THRESHOLD * -1))
    {
        onFaceDown();
    }

    break;
...
```

GRAPH 7. Determining orientation with the gravity sensor. (Milette & Stroud 2012.)

One of the most used sensors in any Android device is the camera. Usually it is used in such a way that the use is just taking photos on its own, but the camera can also be controlled programmatically because it is a built-in sensor. If you want to take a picture with camera via Activity, it isn't hard at all. All that the programmer needs to do is just to create a intent that is going to be started and opens the camera. CODE 4. Shows how it can be done programmatically.

```
private final int PICTURE_ACTIVITY_CODE = 1;
private final String FILENAME = "sdcard/photo.jpg";
private void launchTakePhoto()
{
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    mFile = new File(FILENAME);
    Uri outputFileUri = Uri.fromFile(mFile);
    intent.putExtra(MediaStore.EXTRA_OUTPUT, outputFileUri);
    startActivityForResult(intent, PICTURE_ACTIVITY_CODE);
}
protected void onActivityResult(int requestCode, int resultCode,
                                Intent data)
{
    if (requestCode == PICTURE_ACTIVITY_CODE)
    {
        if (resultCode == RESULT_OK)
        {
            ImageView imageView =
                (ImageView) findViewById(R.id.imageView1);
            Uri inputFileUri = Uri.fromFile(mFile);
            imageView.setImageURI(inputFileUri);
        }
    }
}
```

GRAPH 8. Taking a photo. (Milette & Stroud 2012.)

When talking about the most used sensors in Android devices one should not forget the microphone. Many apps can be used by recording audio data via microphone and then analyze it. As an example app for this is the Android clapper, which has a built in microphone in it and it controls the light of the house. When a person claps his hands, the microphone will record the noise that came out of the clap, and analyze it. After it has been analyzed, the clapper will sent the information to the processor and then the clapper will turn the lights either on or off. The recording of the noise is done by the use of MediaRecorder and AudioRecord APIs. (Milette & Stroud 2012.)

MediaRecorder is the most limited but simplest way to record data to Android device. It uses maximum amplitude which is really useful when you want to detect patterns within an audio

recording. MediaRecorder doesn't have the methods with which it could directly retrieve any raw data, this does make the MediaRecorder an little inconvenience when you are trying to record data and immediately after trying to analyze it. Audio Recording is used when an app needs to do something else at the same time when it is recording the audio. For this to be possible the app needs to record the audio asynchronously and usually the application needs to process the audio data during the same time when it is recorded. This can be done by using the AsyncTask method. AudioRecording has its problems too, when you are trying to analyze raw audio signal there might be some unwanted distortion added in the analyze. So when programmer wants to analyze raw audio data for information that goes beyond the maximum amplitude they should always use AudioRecord instead of MediaRecorder. (Milette & Stroud 2012.)

4.2 iOS

Apple has become well known in all over the world via their Smartphones, iPhones. These equipments from Apple use iOS as their core software. iOS has seen explosive growth since it first came out in 2007. Apple has their own SDK just like Android does. Apple has its own IDE which is called Xcode. It works just like any other IDE. Just like the Android SDK, the Apple SDK has its own simulator also, so you don't actually need to use your own Apple equipment to test if your code or applications actually work. The different thing for developing app's for iOS is that you will need to register to their website to be able to get Xcode and all other necessary information needed for becoming a iOS developer. (Mark, Nutting, LaMarche & Olsson. 2013.)

4.2.1 Sensors in iOS

The availability of sensors in iOS devices is seven. Couples of these sensors are: GPS, accelerometer, magnetometer, and most recently added barometer. There are also some other sensors that are built in the iPhones and used almost every day by iPhone users, such as Ambient light sensor, microphone and camera. These are all included in the iPhone itself, so they are

actually part of the hardware. The accelerometer that is built in iPhone's measures the linear acceleration of the device. This is done so that the accelerometer can report the device's roll and pitch. (Allan 2011.)

One of the main sensors in iOS phone that user can easily handle is the orientation of the device itself. Because every iPhone and iPad have their own so called slider button on the side of the device which can be toggled either on or off. When user does that, the orientation of the device either becomes portrait or landscape depending on the position in which the device has been before the slider button was toggled. Afterwards when the user changed the orientation of the device, it will not actually change the orientation of the screen, because it is now locked and the orientation sensor is not working. (Allan 2011.)

Microphones in iPhones are different from most of the phones. From the era of iPhone 4 Apple has made their phones with three microphones. Where the main microphone is located at the bottom of the phone which is used mostly for normal phone calls, but the second microphone that Apple has placed on their phones is located near their headphone jack. This microphone is intended for video-calling, but it has another purpose too. It is used in conjunction with the main microphone to suppress most of any possible background noises. (Allan 2011.)

4.2.2 Sensor programming in iOS

Camera in iOS devices have been there from the start. First it started just only by having one camera in the backside of the device, but nowadays there are most likely two cameras in every single iOS device, one in the back and one in the front. If a programmer wants to access camera when programming iOS devices via Xcode, he needs to start the program with View-based Application and actually the name of the program needs to be also correct. Program should be named Media. (Allan 2011.)

In iOS SDK you can save images and videos to a Photo Album. This is done by writing a piece of code into the source code, which is `UIImageWriteToSavedPhotosAlbum` and `UISaveVideoAtPathToSavedPhotosAlbum` methods. These both are ways in which you can

save the pictures that you take with your program straight away into the right folders. These two methods are both asynchronous tasks; so if the app is somehow interrupted during filming a video or taking a photo by example a phone call, this will then terminate the filming or photo taking and the image or video will be lost. GRAPH 9 shows an example part of code, which will save an image to the Photo Album using the `UIImageWriteToSavedPhotosAlbum` method. (Allan 2011.)

```
if( [info objectForKey:@"UIImagePickerControllerMediaType"] ==
    kUTTypeMovie ) {

    CGSize pickerSize = CGSizeMake(picker.view.bounds.size.width,
                                    picker.view.bounds.size.height-100);
    UIGraphicsBeginImageContext(pickerSize);
    [picker.view.layer renderInContext:UIGraphicsGetCurrentContext()];
    UIImage *thumbnail = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    imageView.image = thumbnail;

} else {
    UIImage *image =
        [info objectForKey:@"UIImagePickerControllerOriginalImage"];
    UIImageWriteToSavedPhotosAlbum(
        image,
        self,
        @selector(
            imageSavedToPhotosAlbum:didFinishSavingWithError:contextInfo:),

        nil);

    imageView.image = image;
}
[self dismissModalViewControllerAnimated:YES];
}
```

GRAPH 9. Saving image to Photo Album in iOS. (Allan 2011.)

The `AVAudioRecorder` class in iOS devices is part of a framework that provides audio recording capabilities to an app. This is called `AVFoundation`. `AVFoundation` framework will allow the programmer to do the following three steps: record until the user stops the recording, record for a specific duration, and pause and resume a recording. There is a corresponding part of the `AVFoundation` in the `AVAudioRecorder`. This provides quite a few really sophisticated ways in which you can play sound in your applications. This can do the following: Play sounds of any duration, Play sounds from files or memory buffers, Loop sounds, Play multiple sounds simultaneously (one sound only per audio player) with precise synchronization, and control relative playback level and stereo positioning for each sound you are playing, and seek

to a particular point in the sound file, which supports such application features as fast forward and rewind. (Allan 2011.)

By the use of accelerometer with Apple devices, there is a really simple and easy way to determining the orientation of the iOS device. Just a simple call to `UIDevice` will return the current orientation in which the device resides. Orientation of iOS device can be achieved by using accelerometer in the following way. `UIDevice *device = [UIDevice currentDevice];
UIDeviceOrientation orientation = device.orientation;`. (Allan 2011.)

`CLLocationManager` is a class that can be used not only to report current location, but also to report current heading of the device. But this can only be done if the hardware supports it. The location manager also has the possibility to return both true heading and magnetic heading values if location updates have been enabled. If there haven't been any updates enabled the location manager will only return magnetic heading value. Local magnetic field also affects the readings of `CLLocationManager` and usually this is overcome with the calibration of the device. This is usually done by moving it with accordance to figure-eight pattern. If you want to find out the true heading and magnetic heading with their correct values, you should hold your device just like a traditional compass, in portrait mode. If the device is rotated, the heading readings will remain the same, but the person is not changing the direction they are facing, the heading values of the device will have changed because of the rotation of the device. The orientation of the phone will need to be changed before the headings are reported back to the user. (Allan 2011.)

For a programmer to be able to measure the magnetic field with the device for this to be possible, the programmer will need to first calculate the zero-point measurements of the ambient magnetic field of the Earth. All readings afterwards should subtract this zero reading. For a programmer to be able to receive all the necessary raw magnetic field measurements along all the axes X, Y and Z, he needs to query the `CLHeading` object that is passed to the `locationManager:didUpdateHeading:` method. This method and how to include the values in it is shown in GRAPH 10. (Allan 2011.)

```

- (void)locationManager:(CLLocationManager *)manager|
  didUpdateHeading:(CLHeading *)heading {

    double x = heading.x;
    double y = heading.y;
    double z = heading.z;

    double magnitude = sqrt(x*x + y*y + z*z);

    ... code ...
}

```

GRAPH 10. Measuring the magnetic field. (Allan 2011.)

Core Motion framework that was introduced in iOS 4, allowed developers applications to receive data from two different sources at the same time. These two sources were accelerometer and the gyroscope which can be used to measure the rate of change in the direction of the device. The programming part of this is really simple forward, with the `CMMotionManager` class it was now possible to start receiving accelerometer, gyroscope, and combined device motion events at a regular interval. (Allan 2011.)

The `CMMotionManager` class can present the data of accelerometer and gyroscope both separately or as combined `CMDeviceMotion` object. This `CMDeviceMotion` will encapsulate the processed device motion data from both sensors. Because of this the Core Motion will provide the most accurate results in the form of device attitude, rotation of the device, the direction of gravity on the device, and the acceleration that is given to the device by the user. (Allan 2011.)

There are two approaches of how to obtain the data that is given by `CMMotionManager`. Easiest way to do this is to pull the motion data; in this case your application is going to start an instance of the motion manager class and will ask for the combined device motion in periodic format. It is also possible to obtain just the raw data for each one of these sensors that gather the motion data by just calling them like `CMAccelerometerData *accel = motionManager.accelerometerData;`. Most of the developers call this the easiest way to obtain motion data and also the most efficient way. But it has a slight downside in it, if you don't have an natural timer in your app, then you need to add an additional timer to your source code so that it will trigger your update requests. One thing to always to remember is that you need to release the motion manager after you don't need it anymore. (Allan 2011.)

There are other possibilities than just pulling data from motion manager. You can make an update interval and then implement a block of code that will handle all the data collected. The motion manager class can then deliver updates; this can be done by using method named `NSOperationsQueue`. This method allows the handler then to push measurements for the application rather than pulling them to the application. This can be either done to both of the sensors at the same time or individually for either one of them. If the programmer decides to use the second methodology, he will get continuous stream of motion data. But the problem with this method is that it has really large overhead associated with implementing it. (Allan 2011.)

4.3 Windows Phone

Windows Phone 7 was the first Windows Phone that was launched by Microsoft. When Windows Phone 7 came to the market, it had a new application programming model and a huge suite of services that could help programmers and developers to write applications on Windows Phone platform. When they developed the Windows Phone 7, somewhat huge part of their focus was on the developer side as critical part of Windows Phone 7 ecosystem. This have produced a growing commercial opportunity for any software developer who wants to do an application to Windows Phones. (Zhou, Zhu, Zheng & Yang 2011.)

Windows Phone also has their own developer tools for creating an application for Windows Phone. With this SDK comes also an emulator that you can use when you want to actually develop an application and test it without owning an actual Windows Phone. Because this is Microsoft's product, it will be necessary to have either Windows 7 or Windows Vista in the least installed on your computer for you to be able to use this SDK that is required. (Zhou, Zhu, Zheng & Yang 2011.)

4.3.1 Sensors in Windows Phone

As all other Smartphones, Windows Phone also has all the basic hardware sensors that are built in the phone when it is produced. These sensors include the microphone, camera, magnetometer, light sensor, accelerometer, resistive and capacitive touch controller. As the Smartphones have become more and more popular, so has the need for the sensors in Smartphones. More and more people use their mobile devices for communication and entertainment in daily basis. All of these are fulfilled by mobile applications that use sensors in them, and actually one of the main reasons why there has been such an increase in the mobile phone usage for entertainment and other uses is that the mass of new sensors available in today's mobile device is huge. (Zhou, Zhu, Zheng & Yang 2011.)

4.3.2 Sensor programming in Windows Phone

Microphone in Windows Phone 7 (WP7) is one of the most used sensors, and it is more likely to be the easiest one to program by software developers. The simplest way to record audio in WP7 is by using Microsoft's own class for it, which is: `Microsoft.Xna.Framework.Audio.Microphone`. Here is an example source code for recording which is shown in GRAPH 11. First of all you need to add couple assemblies to the code: using `Microsoft.Xna.Framework`; for starters to use the framework and then using `Microsoft.Xna.Framework.Audio`; for being able to use the Audio part of WP7. (Zhou, Zhu, Zheng & Yang 2011.)

```

namespace WP7Microphone
{
    public partial class MainPage : PhoneApplicationPage
    {
        private Microphone myMic = Microphone.Default;
        private byte[] cacheBuf;

        private MemoryStream memoryStorage = new MemoryStream();

        public MainPage()
        {
            InitializeComponent();

            DispatcherTimer dt = new DispatcherTimer();
            dt.Interval = TimeSpan.FromMilliseconds(33);
            dt.Tick += new EventHandler(dt_Tick);
            dt.Start();

            myMic.BufferReady += new
                EventHandler<EventArgs>(microphone_BufferReady);
        }

        void dt_Tick(object sender, EventArgs e)
        {
            try { FrameworkDispatcher.Update(); }
            catch { }
        }

        void microphone_BufferReady(object sender, EventArgs e)
        {
            myMic.GetData(cacheBuf);
            memoryStorage.Write(cacheBuf, 0, cacheBuf.Length);
        }

        void microphone_BufferReady(object sender, EventArgs e)
        {
            myMic.GetData(cacheBuf);
            memoryStorage.Write(cacheBuf, 0, cacheBuf.Length);
        }

        private void recordButton_Click(object sender, EventArgs e)
        {
            myMic.BufferDuration = TimeSpan.FromMilliseconds(1000);
            cacheBuf = new byte[myMic.GetSampleSizeInBytes(myMic.BufferDuration)];
            memoryStorage.SetLength(0);
            myMic.Start();
        }

        private void stopButton_Click(object sender, EventArgs e)
        {
            myMic.Stop();
        }
    }
}

```

GRAPH 11. Example of how to record voice on WP7. (Zhou, Zhu, Zheng & Yang 2011.)

When a software developer wants to access the built-in camera in WP7, they need to use the `Microsoft.Phone.Tasks.CameraCaptureTask` class. When programming an example camera application the developer needs to use `CameraCaptureTask` this is quite simple and straight

forward to use. Whenever the capture button is clicked in the application, the event handler will create a `CameraCaptureTask` object. The data of the picture is going to be carried in the `PhotoResult` object, which is then passed back to the `CameraCaptureTask`'s `Completed` event. When this is done the picture is going to be obtained by the application and it will be saved in the media library of the device. (Zhou, Zhu, Zheng & Yang 2011.)

WP7 applications developed by any programmer can access all the supported sensors in WP7. This can be done by using `Microsoft.Devices.Sensors` namespace. This namespace includes all the sensors that can be accessed via programming. The most used of the sensors in this namespace is accelerometer. It can be used to either retrieve value changes whenever the Windows device is in any motion that includes acceleration. For a programmer to be able to control any data that is going to be collected from the accelerometer sensor, the `Accelerometer` class provides the user with `Start()` and `Stop()` methods. Whenever you want to handle the data you can use the event class `ReadingChanged()`. (Zhou, Zhu, Zheng & Yang 2011.)

When programmer wants to obtain orientation on WP7, there is a class which can be used for it, named `GraphicsDeviceManager` which is part of the `Microsoft.Xna.Framework` namespace. Then in the device manager part of the framework namespace there is a property named `SupportedOrientations` which can be used to get the display orientation or it might be even used by programmers to set the orientation in certain way they want. At the time of this book, there wasn't any other sensors that were supported by the API that was used in WP7. (Zhou, Zhu, Zheng & Yang 2011.)

5 DIFFERENCES IN THE MOBILE PLATFORMS

The largest difference between all three mobile platforms is that they are all developed by their own companies, and they all have their own way of doing their mobile platforms. The three companies are named Apple, Google and Microsoft. Each one of them have their own platform for their own mobile devices, these platforms are following: Apple has MacOS as their underlying operating system and from that they have made their iPhone operating system iOS. Google has Android as their operating system for their mobile platforms which is based on Linux operating system and it is widely used by Samsung and HTC mobile devices. Windows has their own Windows Compact Edition (CE) kernel as their operating system for their mobile platforms.

Other substantial differences between these three mobile platforms are that each one of them has their own application framework with which they work. Apple has their own Xcode for all their iOS devices, Android mobile devices uses Java as the framework, and Windows Phone 7 uses Silverlight or Xbox/DirectX New generation Architecture (XNA) as their application framework. Also all of these mobile platforms have their own app store. Apple has iPhone App Store, Android has Android Market, and Windows Phones have their own Windows Marketplace which is actually also used on other Microsoft devices, such as Xbox. (Zhou, Zhu, Zheng & Yang 2011.)

Biggest difference when comparing these three mobile platforms from the sensor point of view is that Android devices have the most versatile hardware in them, this is because Google doesn't define any hardware specifications to their Android devices. Whereas Apple and Microsoft are more strict about the hardware and all the hardware in their mobile devices are going to be done. So they both have really strict specifications for the hardware included in their mobile devices. (Zhou, Zhu, Zheng & Yang 2011.)

5.1 Difference between the API

Baseline for a good API from programmers view is that it supports as much stuff as possible, and is easy to use for the programmer. When taking this into consideration, comparing these three mobile platforms will show you that the API in Android devices is one of the easiest one. This is mainly because Android API do support quite a few things compared to the other two. Apple is more closed in their API and Windows Phone is catching up with Android, but is not there yet.

Each of the three mobile platforms has their own API. All of them are quite a bit different from each other and all of them do not support all the same features. For example one big difference between the API's is that Apple is very strict with their mobile devices connection to other devices. Their API does not support the communication via Bluetooth without an agreement. Whereas the other two API's do allow their phones to communicate with other devices via Bluetooth without making any kind of agreements in the API.

When talking about sensors and their API supports, Apple is maybe one of the worst ones in that case. This is just because Apple has built their sensors in such a way that they only have one producer for them, where as Android and Windows Phone have both richer sensor supports. Android might be the richest one in this case. This is only because Android allows so many different sensor manufacturers for their sensors, so there is really wide variety of sensor support in their sensor API. The Android sensor object model is also different from the ones in iOS and WP. Android sensor manager has a class which manages all the supported sensors. It is easy to access the sensors in Android devices, mainly because of the uniform access methods of Android API. (Allan 2011.)

One thing that Windows Phone API beats the other two mobile platforms API's is the support of FM radio. Neither iOS nor Android provides any kind of support for FM radios in their API. This is one thing that Microsoft phones and before that Nokia phones have always supported. It makes them a little bit different in that way from the other two mobile platform devices. Windows Phone devices usually come with built-in FM radio chip. One thing to re-

member that they are lowering the support for FM radio all the time and the newest Windows Phones do not actually have FM radio chip in them anymore. (Zhou, Zhu, Zheng & Yang 2011.)

5.2 Difference between programmability

First thing discussed when talking about programmability is the architecture of the respective OS's. Windows OS contains three layers in it as shown in TABLE 3: Hardware, Kernel space and User space. All the .NET Framework applications run in the user space and the entire OS kernel, all the possible drivers and system services run in the Kernel space, and finally all the sensors and other hardware oriented stuff reside in the hardware layer. (Zhou, Zhu, Zheng & Yang 2011.)

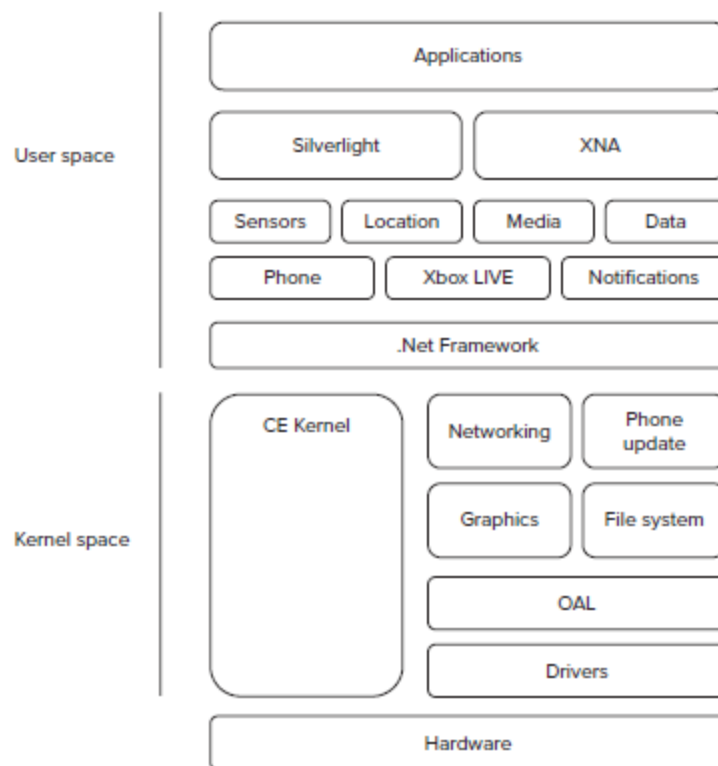


TABLE 3. WP7 architecture. (Zhou, Zhu, Zheng & Yang 2011.)

Android architecture is built in such a way that all the drivers and software is based on the Linux Kernel as is shown in TABLE 4. In the Android architecture the Linux kernel comes the Dalvik virtual machine. This machine is a special Java virtual machine that is optimized to run Java applications on mobile devices. In Android majority of its operating systems are in the application framework, which is mainly consisted of Java services, API's, and libraries as well as native libraries. (Zhou, Zhu, Zheng & Yang 2011.)

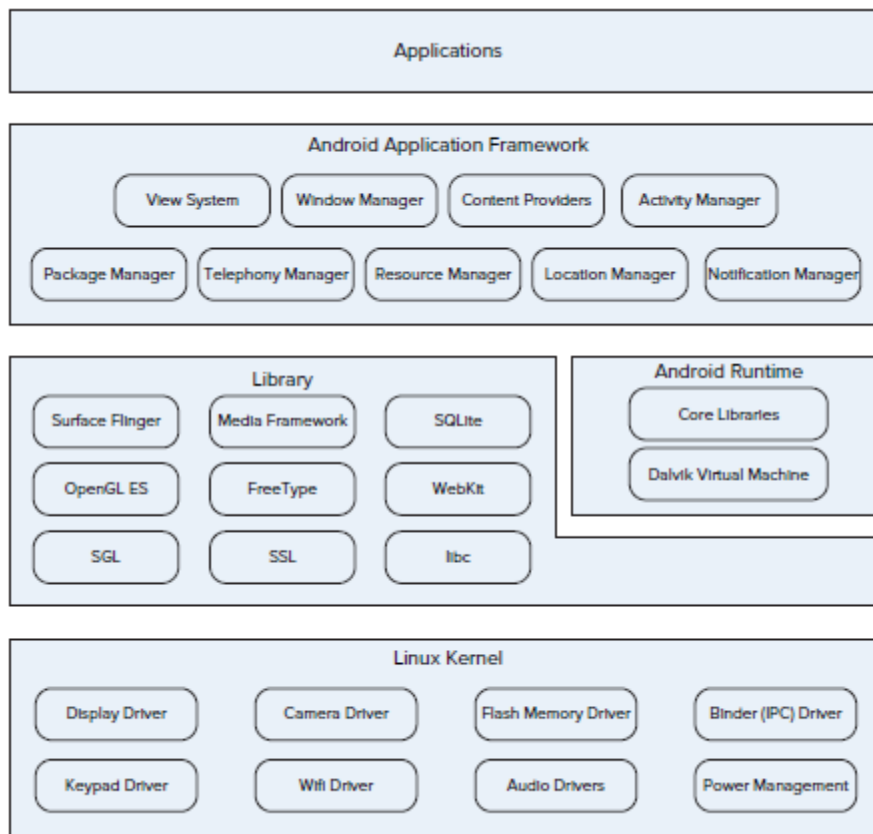


TABLE 4. Android architecture. (Zhou, Zhu, Zheng & Yang 2011.)

Finally the third architecture is iOS architecture. It acts as an intermediary between all the hardware that is in iOS devices and the applications that developers will create. The applications in iOS will not talk to the hardware directly in any case, but to do this connection they usually communicate through well-defined system interfaces. Because of these interfaces, it is easy to write apps that are going to work consistently on devices that are having hardware capabilities. The way that Apple is showing their implementation of iOS technologies is in way or layers. Lower layers contain the fundamental services and technologies included in the iOS.

Whereas the higher-level layers are build upon the lower layers and therefore they provide more sophisticated services and technologies. (iOS Developer 2014.)

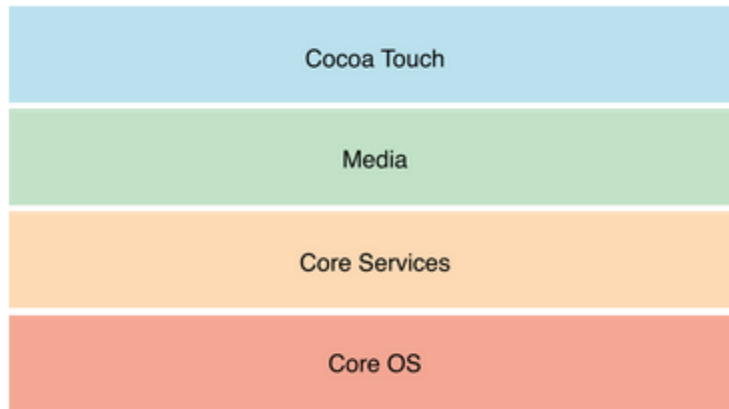


TABLE 5. Layers of iOS. (iOS Developer 2014.)

As shown in the TABLE 3, 4 and 5 that three of these mobile platforms have their own unique way in which their architecture works. The comparison between these three architectures can be based on the complexity they are having in them. By looking at the three different architectures from programmers point of view, the most easiest one is most likely going to be iOS, this is because the architecture build for iOS is pretty simple and straight forward, whereas the architecture for Android and WP7 is more complex in the way the layers are communicating with each others. The most complex of these three would be most likely Windows Phone architecture; this is because the Windows Phone architecture has the most complex way in which the layers work, such as the user space layer in WP7. It includes all of .NET Framework, which makes it more complex than the two other architectures which are going in chronological order from programmer's point of view.

Frameworks are also important when talking about the differences between the programmability of these three mobile platforms. All of the iOS app's are built on top of the Cocoa Touch layer of their architecture. This layer consists of Objective-C application frameworks, such as UIKit, Game Kit, Map Kit, and iAd. The Cocoa Touch layer of the iOS devices provides a really good amount of UI objects and event-handling mechanisms for applications and core system services. Usually developers will use functions that are already provided by the Cocoa

Touch frameworks. Even though it is possible to call directly to the core service layers, but it is much easier to do it by using Cocoa Touch framework. (iOS Developer 2014.)

Just like iOS applications, the Android Java applications are built on top of Android framework. This framework consists of an array of Java packages that are used. These packages are: Activity Manager, View System, Windows Manager, Telephony Manager, Content Provider, Notification Manager, and so on. The Android UI is usually composed of a variety of different view components which are then used in combination of layouts. The framework's Java packages are the so called interface to all the underlying system capabilities. This is because they rely on corresponding native libraries to perform some of the lower-level tasks in the architecture. One thing that Android does differently is that they have introduced some programming concepts that make it easier to extend system applications for developers. These concepts are Activity and Intent. (Zhou, Zhu, Zheng & Yang 2011.)

The application framework for WP7 is .NET-based. In WP7 there are no possibilities to use native code. The frameworks Silverlight and XNA do both provide a certain set of .NET Framework types. These framework types usually serve as interface to the underlying OS. Windows Phone API makes it possible for applications to interact with media, local and remote data, and also sensors. On WP7 the .NET framework runs on a sandbox, this means that no cross-application communication is allowed on the WP7. So if a programmer wants to do cross-app communication they have to rely on web services as a bridge for it. (Zhou, Zhu, Zheng & Yang 2011.)

When comparing the app framework for these three mobile platforms you can see some differences in them all. There are couple big differences when comparing these. One of them is that as shown in the table, iOS and Android only have one application framework, whereas WP7 has two. All of these three frameworks have their own programming language, iOS is programmed in Object-C, Android is programmed in Java and WP7 is programmed in C# and Visual Basic.NET. All of them have their own SDK also, this makes the programmers life quite hard if he wants to be able to code all three of these. TABLE 6 shows the basic application frameworks for each of these mobile platforms. One of the main reasons for this is that

iOS SDK doesn't work well in Windows computers and Windows Phone SDK doesn't work well in Mac OS computers. The third and last of the big differences between these frameworks is that iOS and Android allow native API access, but WP7 doesn't allow any native API access. This makes it less practical than the other two. (Zhou, Zhu, Zheng & Yang 2011.)

ITEM	IOS	ANDROID	WP7
Available application frameworks	Cocoa Touch	Android framework	Silverlight and XNA
Programming languages	Object-C	Java	.NET languages (C# and Visual Basic.NET)
Native API access	Yes	Yes, via JNI using NDK	No
SDK	iOS SDK on Mac OS X	Android SDK on Windows, Linux and Mac OS X	Windows Phone SDK on Windows 7
IDE	Xcode	Eclipse (with ADT plug-in)	Visual Studio 2010 or Visual Studio 2010 Express
UI definition	NIB resource file	Widgets and layouts	XAML
UI event mechanism	Delegates	Event listeners in View classes	Event handlers in CLR (Common Language Runtime)

TABLE 6. Application framework comparison table. (Zhou, Zhu, Zheng & Yang 2011.)

Because all of these mobile platforms have their own coding language used in them, the accessing of data in sensors is different for each one of them. For example getting data from accelerometer differs in each one of them. GRAPH 12 is an example from Windows phone coding, GRAPH 13 is from iOS and GRAPH 14 is Android version. All of these are doing the same thing, but they all differ because of differences in coding language. (Zhou, Zhu, Zheng & Yang 2011.)

```

public void AccelerometerReadingChanged(object sender,
    AccelerometerReadingEventArgs e)
{
    accelReading.X = (float)e.X;
    accelReading.Y = (float)e.Y;
    accelReading.Z = (float)e.Z;
}

```

GRAPH 12. Obtain data from accelerator in WP. (Zhou, Zhu, Zheng & Yang 2011.)

```

- (void)accelerometer:(UIAccelerometer *)accelerometer didAccelerate:(UIAcceleration *)acceleration
{
    UIAccelerationValue x, y, z;
    x = acceleration.x;
    y = acceleration.y;
    z = acceleration.z;

    // Do something with the values.
}

```

GRAPH 13. Obtain data from accelerator in iOS. (iOS Developer 2014.)

```

public void onReadAccelerationDataToggleButtonClicked(View view)
{
    ToggleButton toggleButton = (ToggleButton)view;

    if (toggleButton.isChecked())
    {
        startReadingAccelerationData();
    }
    else
    {
        stopReadingAccelerationData();
    }
}

```

GRAPH 14. Obtain data from accelerator in Android. (Zhou, Zhu, Zheng & Yang 2011.)

5.3 Difference in designer/developer guidance between platforms

All three mobile platforms have their own website, which can be used to find out information about developing any apps or designing apps. Their websites can also be used to search for any information that programmer wants to know. For example if a programmer is developing an app which has buttons in it and you want to know which commands a person needs to use to do buttons, you can go to the mobile platform's developer website and from there anyone can search for the information that a person is going to need.

Android has their developer website in which you can find all information necessary for you to develop any app you want. One thing that Android has that other places do not support is

that you can find any information you want in the Android Developer website without registering to the website or to any Android program. Another thing that Android developer website does really well is that it is actually easy to use for starters; it has its own Get Started place which is going to help you get started in Android programming. This Get Started place has example code pieces for everything that they ask you to do and they show you how you can do it and also they do explain those things pretty well in my opinion. One thing that Android developer website lacks in my opinion is that when you go to the deeper parts of coding, they only have small samples of how to do something and that's it. (Android Developer 2014.)

Windows Developer website also is filled with information of anything you need to actually program an application. Only thing that Microsoft wants you to sign up at is that if you want to publish your application to the Windows Phone Store. The guidance inside Windows Developer website is really well done, and it actually has a lot of code examples in there for almost anything you want to search for. For example if you search for how to display orientation given by accelerometer and gyroscope it has detailed guidance how you can do it and it also has an example code in there which you can either use yourself or modify little bit to your liking. They also have large library from where you can search for almost anything you want to find out about some coding objects or namespaces and so on. (Windows Dev Center 2014.)

Apple and iOS is the only one of the three mobile platforms that requires the developer to first of all to register and sign in to Apple Development website. If you don't register you will not be able to access all the available information from their website and the guidance is really poor at that point. But if you do decide to sign up, which is free and easy to do. When you go to the Developers library part of Apple's website, you will find out that finding certain information can be quite challenging because it is not the easiest place to use. But when you actually have found the information that you are looking for, the guidance for that part is really good and easy to understand. One thing Apple is lacking is that they don't have as much of sample codes in their website as Windows Developers and Android Developers have. But Apple has done really well the explaining of any object you want to search for. For example the explanation of CMMotionManager is done in such a way that it is easy to understand what it does. (iOS Developer 2014.)

So to compare these three websites the Windows Developer website is one of the easiest one to use and find out information about. It also has the most amounts of sample codes in there and the sample codes are also explained in there. Android has the second most amounts of sample codes included in their explanation, but the code samples found from Android Developer site are not always as simple as they could be. Apple has the best explanation of their objects and so forth compared to the other two developer websites.

5.4 Difference between usability

All of these three different mobile platforms have their own way in which they are used by users. Also the usability is one of the key things when talking about today's mobile devices. If the device is too hard to use, most likely it will not have that many users. Even though all three mobile platforms have their own style, they are still quite a bit similar to each other. Because in the end they all have the same needs that they are used for and most likely they will all work in a same way but with their own unique style.

Android devices usability has always been the same style. They have had their start screen as simple as possible from the beginning, but still they have all the necessary parts to make it useful in the end. When a user presses the menu button from start screen, the menu will open which then has everything in it. This in my opinion is one of the flaws in Android, because it is most likely going to be quite a mess when you open the menu. This is if you have a lot of apps on the phone. Otherwise Android devices are simple and clean in every way.

Windows mobile devices have always been almost the same way as their Windows OS for computers. Most recent example of this is Windows 8. The start screen in Windows 8 is same looking and working in the same way as the Windows 8.1 version that is on mobile devices today. The usability of this style is quite similar in both ends, so if the user knows how to use one of Microsoft's devices he can then use the other one also. The flaw in Windows mobile devices is that the so called app menu, where everything is does look too similar to the start screen to which you can actually pin everything if you like. It is actually really simple to use,

but for users it might not look as nice as the other mobile platforms. This is because if the user has too much stuff on his/her start screen it is quite a mess.

Finally the iOS devices have their own unique way in which Apple has it designed. It is most likely the easiest one to use, this is mainly because one of the main points of Apple is actually make the phone really easy to use for their users. In Apple devices there is a possibility to flip the screen from one to another where apps are located. There is different app for all the settings etc. Also one important and really useful part that Apple has is that there is a possibility to make a folder of your apps. For example name it Games and put all games in there. Programmer can do this to almost every single app, so the way that Apple has designed this is really simple and smooth looking for the users also.

So when comparing these three mobile platforms from their usability point of view; Apple is the easiest one to use by any user, beginner or experienced user. Windows mobile devices come second and finally come the Android devices. This is only because Windows has done it in such a way that if you actually have Windows 8 on your computer, it is quite similar to use. Android and Windows mobile devices are tied and this is the only reason Windows is before Android on the usability list. Android also might be the hardest one to start with, but when you know how to use it is simple. But this is true for all the mobile platforms.

6 CONCLUSIONS

The primary goal of this thesis was to discuss about sensors and how they are accessed on nowadays mobile platform devices. How can a programmer that has never programmed mobile platforms do it, and how can a person who has never been told about any sensors in the mobile phone realize that there actually are large amount of sensors embedded in today's mobile platform devices. Also there is comparison of three of the most used mobile platforms which in this case are Android, iOS and Windows Phone.

Sensors are part of mobile devices today and there is a large amount of them. The most commonly used sensors are cameras and microphones which most people do not even recognize as sensors. This is because they are used every day by people who have mobile devices and therefore they do not think of them as sensors. These two sensors can be used to collect large amounts of data. From programmers point of view almost any sensor in a mobile device is really a useful one. This is because one single sensor can be used to collect certain data that the programmer actually wants, or then a programmer can use two or even three sensors to obtain certain data correctly. An example of this is the use of accelerometer and magnetometer in combination to obtain the orientation of the mobile device.

Some sensors that are used by people and they might not even know about it are location services such as GPS. This happens mainly because GPS is used by many applications to find out your exact position. Such as weather applications want to use the location of the smartphone user to find out the nearest weather station and then show the weather that is going to be in the area of the smartphone user. GPS is also used by many sports applications, the data gathered by the GPS can be used by the programmer to actually find out the location in which the phone has been in.

When comparing the most used mobile platforms, Android, iOS, and Windows Phone there are many differences in them. As programmer the main difference between these three mobile platforms is that they are all programmed in different OS, and they all have their own API. The difference in the API might be a quite difficult thing for a programmer. This is because if

the programmer wants to be good in programming all of the three mobile platforms he needs to know large amount of code and how the API's work. More notable differences in these mobile platforms are that there might not be support for a certain sensor API, for example iOS and Android do not support FM radio, where as WP does. There are differences in the programmability of these mobile platforms also, as they all have their own framework in which they are based on and if a programmer wants to be good at programming these three platforms, he should know how the frameworks for each of the platform works.

So for programmers the sensors of mobile devices are really good and useful embedded systems. They can obtain large amounts of data by using them. But it might be hard to master all three of these. When looking at the comparison part of this paper, one can notice that WP might be the most beginner friendly one, then comes Apple for beginners, and finally Android. But if a programmer has worked with mobile platform programming before, Android might be the easiest one to program. Android has the most chronological order in its code and how to program it and it also has the largest support for sensors of the three mobile platforms. So from mobile programmers point of view Android might be the easiest one to use and program on.

For further study one could research to what extent could smartphone sensors be used for pure data gathering? Could the sensors that people use every day be easier for them to actually use and how that could be possible to achieve. How to make sensors simpler for people who are not programmers to understand, how to make the user interface of certain sensor applications easier to use and how to make the coding of sensors easier for everyone.

REFERENCES

Allan, A. 2011. Basic Sensors in iOS. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Android Developer. 2014. Sensor. Wwww-document.

Available: <http://developer.android.com/reference/android/hardware/Sensor.html>

Accessed on 18.11.2014.

Aram, S., Troiano, A., Pasero, E. 2012. Environment Sensing using Smartphone. IEEE Sensors Applications Symposium (SAS) 2012, 1-4.

Burnette, E. 2010. Hello, Android: Introducing Google's Mobile Development Platform Third Edition. Pragmatic Bookshelf. Dallas, Texas.

Bray, J., Struman C.F. 2001. Bluetooth 1.1: Connect Without Cables (2nd Edition). Prentice Hall. United States of America.

Gardner, N., Haeusler, H.M., Tomitsch, M. 2010. Infostructures: A Transport Research Project. Freerange Press, Sydney, Australia.

Gupta, N. 2013. Inside Bluetooth Low Energy. Artech House. United States of America.

Hernset, O., Boswarthick, D., Elloumi, O. 2011. Internet of Things: Key Applications and Protocols (2nd Edition). John Wiley & Sons, Inc., Indianapolis, Indiana.

iOS Developer. 2014. iOS Developer Library. Available: <https://developer.apple.com/> Accessed on 20.11.2014.

Lane, N.D., Miluzzo E., Lu H., Peebles D., Choudhury T., and Campbell A.T. 2010. A Survey of Mobile Phone Sensing. IEEE Communications Magazine 48, 140-150.

Mark, D., Nutting, J., LaMarche, J., Olsson, F. 2013. Beginning iOS6 Development: Exploring the iOS SDK. Apress Media, LLC, California.

Milette, G. & Stroud, A. 2012. Professional Android Sensor Programming. John Wiley & Sons, Inc., Indianapolis, Indiana.

Windows Dev Center. 2014. Development Center. Wwww-document.

Available: <http://dev.windows.com/en-us>

Accessed on 20.11.2014.

Zhou, Z., Zhu, R., Zheng, P., Yang, B. 2011. Windows Phone 7 Programming for Android and iOS Developers. John Wiley & Sons, Inc., Indianapolis, Indiana.